Manual de Usuario

Introducción

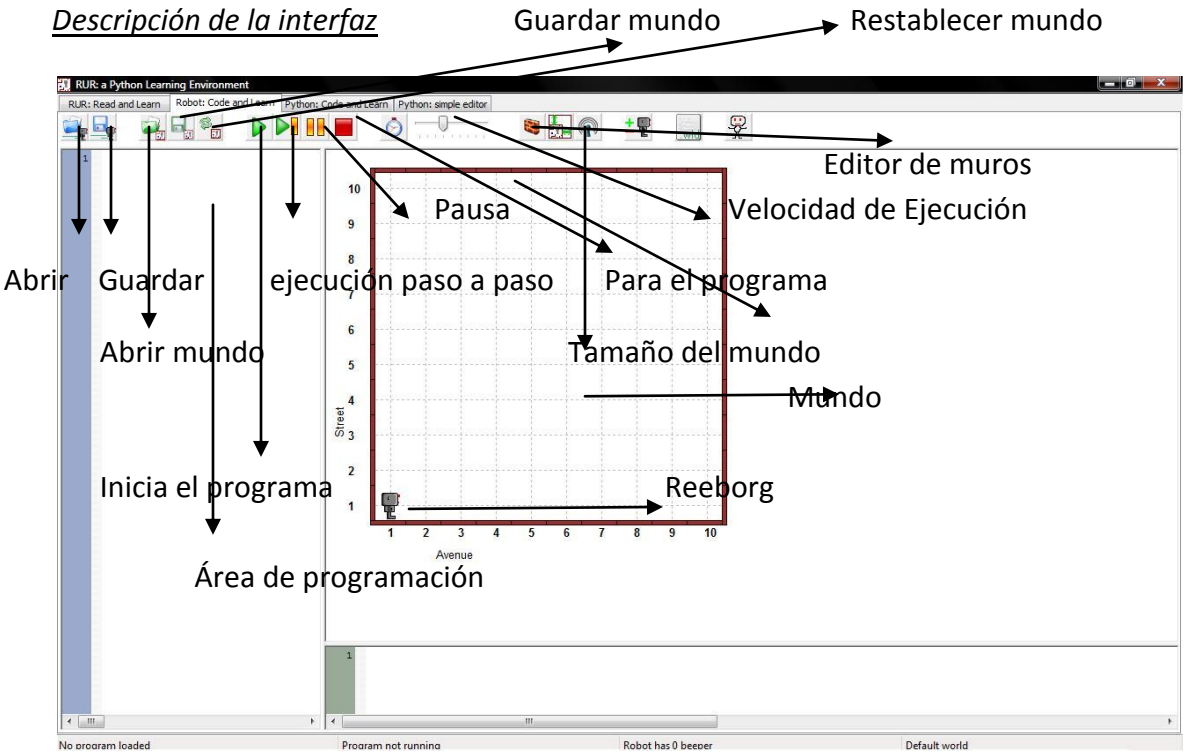
Empezaremos por definir que es RUR-PLE. RUR-PLE es un entorno para facilitar el aprendizaje para programar utilizando el lenguaje de programación Python. Cuenta con una interfaz dinámica que consiste en un rectángulo rodeado por paredes. Las líneas horizontales se llaman avenidas y las líneas verticales se llaman calles. Por estas calles y avenidas se desplaza un robot llamado Reeborg. Reeborg es el encargado de ejecutar todas las instrucciones que se le indiquen, como moverse a través del mundo, recoger beepers que son indicadores que Reeborg puede detectar, los puede agarrar y los puede volver a poner, esquivar paredes, no puede chocar con las paredes y un sinfín de actividades más.

Reeborg se mueve

Reeborg cuenta con 3 comandos que puede ejecutar. El resto es cuestión de tu imaginación para combinar estos comandos con condiciones, repeticiones y ciclos para lograr lo que sea. Estos tres comandos son:

- *move()* sirve para que Reeborga avance una sola vez.
- *turn_left()* Reeborg solo puede girar hacia la izquierda, así que tendrás que jugar con este comando para que Reeborg avance.
- *turn_off()* Reeborg necesita apagarse cuando termine sus instrucciones. Este comando apaga a Reeborg.

Descripción de la interfaz



Repeticiones, Ciclos y Condiciones

Como se menciono, Reeborg solo puede ejecutar 3 órdenes. A continuación se describirán como combinar esas 3 órdenes para que Reeborg haga lo que sea.

La instrucción *while* nos sirve para "mientras haga algo pase o realice esto" es una condición, y es muy común a la hora de toparnos con muros. Reeborg despliega un mensaje de error a la hora de chocar con un muro, por lo que se coloca el while y la instrucción de cuando no haya muro que avance. Por ejemplo

```
while front_is_clear():
    move()
```

Esto define que Reeborg se va a mover al frente siempre y cuando no haya muro. Y existe la contraparte el *while not* que es lo contrario al *while.*

La siguiente condición es el *if* es la típica condicional, si tenemos una situación y es verdadera se dar un proceso pero si es falsa será otro proceso. Para esta condicional existen sus contrapartes *elif* y *else.* Que reflejan "de lo contrario…"

Los ciclos son un conjunto de instrucciones definidas, para esto se utiliza el comando *def* y esto lo que hace es vincular varias instrucciones para ejecutarlas como una sola más adelante del programa. Un ejemplo seria si queremos cruzar a la derecha.

```
def derecha():
    turn_left()
    turn_left()
    turn_left()
```

Entonces más adelante en el programa escribiríamos solo *derecha()* para referirnos a girar hacia la derecha.

Las repeticiones son muy útiles en casos que tengamos la misma instrucción varias veces. Estas se pueden simplificar en una sola línea y no importa cuántas repeticiones tengan que tener. Por ejemplo en el ejemplo anterior tenemos 3 turn_left() eso se puede simplificar mediente repeat("la instrucción",# de veces)

```
def derecha():
    repeat(turn_left,3)
```

A partir de estos comandos e instrucciones estarás listo para programar en RUR-PLE, a continuación explicaremos el código de nuestro programa, Boliche, algunas partes no están descritas como función anteriormente porque no son universales si no solo aplican en este programa.

Reeborg juega boliche

(*Consultar Código fuente*)

Se definió el nombre de las variables para poderlas utilizar con un nuevo nombre.

avanti = move

destra = turn_left()

Se definió el contador con el nombre de bolos, para poder identificar cuantos bolos se pide que se agarren.

Bolos=0

Definimos derecha() para los giros hacia la derecha del Reeborg, y luego procedimos definir la variable norte para que Reeborg pueda quedar en posición para observar el norte.

Empezamos a definir nuestras funciones se utiliza *while not* que mientras no esté en posición norte, gire y se coloque en posición, para luego avanzar. Se coloco un *if* debido a que si gira pero y hay pared enfrente marcaria error.
Se coloco un *elif* por si la condición de la variable *if* de arriba no se cumple se pasa a la siguiente instrucción.

Se definió avanzar para lograr que subiera de calle en calle debidamente y se coloco norte para que se ubicara Reeborg. Luego se colocan varios *if* para especificar que el robot no se vaya a estrellar en ningún muro. Luego se colocan varios *elif* para tener diferentes opciones si no se cumplen las condiciones del *if*.

Luego procedemos a definir nuestro contador el que va analizar el beeper y para que Reeborg vaya a recoger los beepers que le dice.

global bolos

Se utiliza global bolos para definir el contador en bolos como variable que se va a utilizar en todo el ciclo.

while on_beeper():

Se utiliza *while* para establecer que mientras encuentre beeper realice el proceso.

pick_beeper()

bolos+=1

Se definió *bolos +=1* debido a que nuestro contador aumentara unos conforme agarre los beeper. Definimos recoger porque el robot después de contar los beepers que agarro de primero, debe ir a recoger esa cantidad. Entonces avanza por condiciones y que los beeper estén encendidos.

Se procede a definir cómo va a recoger los bolos contados. Mientras haya beepers encendidos Reeborg seguirá buscando, pero si se topa con muros regresa a las condiciones para esquivarlo.

Luego hacemos que Reeborg busque los beepers que el contador le dice que agarre. Definimos y agregamos que si hay muro entonces que se mueva.

if bolos > 0:

Condicionamos nuestra variable para que cuando Reeborg detecte que hay que botar los bolos avance y vaya a recoger.

avanzar()

Definimos ir que significa cuando el numero de bolos se mas mayor a 0 entonces si hay beepers Reeborg los va a recoger.

global bolos

Definición de nuestra variable, siempre se define la variable para evitar errores.

if bolos > 0:

Condición de que el numero de bolos debe ser mayor.

if on_beeper():

Si el beeper esta encendido entonces reeborg lo va a traer.

pick_beeper()

bolos = bolos -1

Resta 1 para que el contador vaya disminuyendo, porque cuando llegue a 0 ya no va a seguir agarrando beepers.

while bolos>0:

Definimos que mientras los bolos sean mayores a 0 Reeborg va a seguir buscando

buscar()

while bolos>9:

Reeborg solo puede derribar 9 bolos, es por eso que cuando excede esa cantidad, Reeborg se apaga.

turn_off()

norte()

Reeeborg se queda viendo al norte para finalizar el programa.

Turn_off()

Fin del programa.